Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи №6
з навчальної дисципліни «Методи наукових досліджень»

Тема:
ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ

Виконав:
Студент групи ІВ-92,
Карпека Дмитрій Юрійович

Перевірив:
Регіда П. Г.

Київ 2021

Мета роботи: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи **рототабельний** композиційний план.

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ *l*; - *l*; 0 для $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

Варіант:

210	-25	-5	-70	-10	-25	-5	4,6+5,6*x1+7,0*x2+3,9*x3+1,6*x1*x1+0,7*x2*x2+0,5*x3*x3+9,6*x1*x2+0,6*x1*x3+2,5*x2*x3+3,7*x1*x2*x3

Виконання роботи:

1) Результати роботи програми:

Див. вкладений документ result.pdf

Код програми:

```
from random import randint, uniform
import prettytable
import math
from scipy.stats import f, t
from sklearn import linear_model
from functools import partial

# Лабораторна робота №6 "Проведення трьохфакторного експерименту при
використанні рівняння регресії
# з квадратичними членами" з предмету МОПЕ
# Варіянт №210 Карпека Дмитрій

variant = 210

min_x = [-25, -70, -25]
max_x = [-5, -10, -5]

x0 = [1]
norm_x= [[-1, -1, -1],
         [-1, -1, 1],
         [-1, 1, -1],
         [-1, 1, 1],
         [1, -1, -1],
         [1, -1, 1],
         [1, 1, -1],
         [1, 1, 1],
         [-1.73, 0, 0],
         [1.73, 0, 0],
         [0, -1.73, 0],
         [0, 1.73, 0],
         [0, 0, -1.73],
         [0, 0, 1.73]]
```

```
x0i = list(map(lambda xmax, xmin: (xmax + xmin)/2, max_x, min_x))
delta_xi = [xmax - xi for xmax, xi in zip(max_x, x0i)]

natur_x = [ [min_x[0], min_x[1], min_x[2]],
            [min_x[0], min_x[1], max_x[2]],
            [min_x[0], max_x[1], min_x[2]],
            [min_x[0], max_x[1], max_x[2]],
            [max_x[0], min_x[1], min_x[2]],
            [max_x[0], min_x[1], max_x[2]],
            [max_x[0], max_x[1], min_x[2]],
            [max_x[0], max_x[1], max_x[2]],
            [round(-1.73*delta_xi[0]+x0i[0], 3), x0i[1], x0i[2]],
            [round(1.73*delta_xi[0]+x0i[0], 3), x0i[1], x0i[2]],
            [x0i[0], round(-1.73*delta_xi[1]+x0i[1], 3), x0i[2]],
            [x0i[0], round(1.73*delta_xi[1]+x0i[1], 3), x0i[2]],
            [x0i[0], x0i[1], round(-1.73*delta_xi[2]+x0i[2], 3)],
            [x0i[0], x0i[1], round(1.73*delta_xi[2]+x0i[2], 3)]
            ]

def experiment(m=3, n=14):

    regression_str = 'y = {} + {} * x1 + {} * x2 + {} * x3 + {} * x1x2 + {} *
x1x3 + {} * x2x3 + {} * x1x2x3 + {} * x1**2 + {} * x2**2 + {} * x3**2'

    def func(x1, x2, x3):
        return 4.6 + 5.6*x1 + 7.0*x2 + 3.9*x3 + 1.6*x1*x1 + 0.7*x2*x2 +
0.5*x3*x3 + 9.6*x1*x2 + 0.6*x1*x3 + 2.5*x2*x3 + 3.7*x1*x2*x3

    def matrix_plan(m, n):
        return [[(func(natur_x[0][i], natur_x[1][i], natur_x[2][i]) + uniform(0,
10) - 5) for i in range(m)] for _ in range(n)]

    def multiplication(a, b):
        return a * b

    def average(list):
        return sum(list) / len(list)

    def round_to_2(number):
        return round(number, 2)

    def dispersion(list_y, aver_list_y):
        return [round_to_2(average(list(map(lambda y: (y - aver_list_y[i]) ** 2,
list_y[i])))) for i in range(len(list_y))]

    def cochrane_criteria(S_y):
        global m
        print("\nКритерій Кохрена\n")
        Gp = max(S_y) / sum(S_y)
        q = 0.05
        q_ = q / f2
        chr = f.ppf(q=1 - q_, dfn=f1, dfd=(f2 - 1) * f1)
        Gt = chr / (chr + f2 - 1)
        print("Тест Кохрена: Gr = " + str(round(Gp, 3)))
        if Gp < Gt:
            print("Дисперсії однорідні з імовірністю 0.95")
            pass
        else:
```

```
            print("\nДисперсії неоднорідні.\nПовтор експерименту для m + 1\n")
            m = m + 1
            experiment(m)

    def student_criteria(S_y, d):
        print("\nКритерій Ст'юдента\n")
        bettaList = [sum(S_y) * x0[0] / n,
                     average(list(map(multiplication, S_y, x1i))),
                     average(list(map(multiplication, S_y, x2i))),
                     average(list(map(multiplication, S_y, x3i))),
                     average(list(map(multiplication, S_y, norm_x12))),
                     average(list(map(multiplication, S_y, norm_x13))),
                     average(list(map(multiplication, S_y, norm_x23))),
                     average(list(map(multiplication, S_y, norm_x123))),
                     average(list(map(multiplication, S_y, norm_sq_x1))),
                     average(list(map(multiplication, S_y, norm_sq_x2))),
                     average(list(map(multiplication, S_y, norm_sq_x3)))]
        bettaList = [round_to_2(i) for i in bettaList]

        list_t = [bettaList[i] * S for i in range(n-3)]

        for i in range(n-3):
            if list_t[i] < t.ppf(q=0.975, df=f3):
                list_b[i] = 0
                d -= 1
                print('Коефіцієнт b' + str(i) + ' незначимий, тому виключається
із рівняння регресії')
        print("\nСкореговане рівняння регресії:")
        print(regression_str.format(*map(round_to_2, list_b)))

    def fisher_criteria(d):
        global m
        print("\nКритерій Фішера\n")
        f4 = n - d
        S_ad = (m * sum(
            [(list_b[0] + list_b[1] * x1i[i] + list_b[2] * x2i[i] + list_b[3] *
x3i[i] + list_b[4] * norm_x12[i] +
             list_b[5] * norm_x13[i] + list_b[6] * norm_x23[i] + list_b[7] *
norm_x123[i] + list_b[8] * norm_sq_x1[i] + list_b[9] * norm_sq_x2[i] +
list_b[10] * norm_sq_x3[i]
             - average_list_y[i]) ** 2 for i in range(n)]) / f4)
        Fp = S_ad / Sb

        if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3):  # перевірка критерію Фішера з
використанням scipy
            print('Математична модель неадекватна експериментальним даним на
рівні значимості 0.05.\nПовтор експерименту для m+1')
            m = m + 1
            experiment(m)
        else:
            print('Математична модель адекватна експериментальним даним на рівні
значущості 0.05')

    def printed_matrixes():
        pt1 = prettytable.PrettyTable()
        pt2 = prettytable.PrettyTable()
        pt1.field_names = ["X0", "X1", "X2", "X3"] + ["X12", "X13", "X23",
"X123"] + ["X1**2", "X2**2", "X3**2"] + ["Y" + str(x) for x in range(1, m + 1)]
+ ["Aver Y"] + ["S_y"]
```

```
        pt2.field_names = ["X0", "X1", "X2", "X3"] + ["X12", "X13", "X23",
"X123"] + ["X1**2", "X2**2", "X3**2"] + ["Y" + str(x) for x in range(1, m + 1)]
+ ["Aver Y"] + ["S_y"]

        print("Матриця повного факторного експерименту з натуралізованими
значеннями:\n")
        pt1.add_rows([x0 + natur_x[i] + natur_x12[i] + natur_x13[i] +
natur_x23[i] + natur_x123[i] + [natur_sq_x1[i], natur_sq_x2[i], natur_sq_x3[i]]
+ matrix_y[i] + [average_list_y[i]] + [S_y[i]] for i in range(n)])
        print(pt1)

        print("\nМатриця повного факторного експерименту з нормалізованими
значеннями:\n")
        pt2.add_rows([x0 + norm_x[i] + [norm_x12[i]] + [norm_x13[i]] +
[norm_x23[i]] + [norm_x123[i]] + [norm_sq_x1[i], norm_sq_x2[i], norm_sq_x3[i]] +
matrix_y[i] + [average_list_y[i]] + [S_y[i]] for i in range(n)])
        print(pt2)

    m = m
    n = 14

    x_average_max = sum(max_x) / 3
    x_average_min = sum(min_x) / 3

    y_max = round(200 + x_average_max)
    y_min = round(200 + x_average_min)

    matrix_y = matrix_plan(m, n)
    average_list_y = [round(average(matrix_y[i]), 2) for i in
range(len(matrix_y))]

    S_y = dispersion(matrix_y, average_list_y)

    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    d = 11

    Sb = sum(S_y) / n
    S = math.sqrt(Sb / (n * m))

    norm_x12 = [round(norm_x[i][0] * norm_x[i][1], 3) for i in
range(len(norm_x))]
    norm_x13 = [round(norm_x[i][0] * norm_x[i][2], 3) for i in
range(len(norm_x))]
    norm_x23 = [round(norm_x[i][1] * norm_x[i][2], 3) for i in
range(len(norm_x))]
    norm_x123 = [round(norm_x[i][0] * norm_x[i][1] * norm_x[i][2], 3) for i in
range(len(norm_x))]
    #--
    norm_sq_x1 = [round(norm_x[i][0]**2, 3) for i in range(len(norm_x))]
    norm_sq_x2 = [round(norm_x[i][1]**2, 3) for i in range(len(norm_x))]
    norm_sq_x3 = [round(norm_x[i][2]**2, 3) for i in range(len(norm_x))]
    #--

    natur_x12 = [[round_to_2(natur_x[i][0] * natur_x[i][1])] for i in
range(len(natur_x))]
    natur_x13 = [[round_to_2(natur_x[i][0] * natur_x[i][2])] for i in
range(len(natur_x))]
```

```
    natur_x23 = [[round_to_2(natur_x[i][1] * natur_x[i][2])] for i in
range(len(natur_x))]
    natur_x123 = [[round_to_2(natur_x[i][0] * natur_x[i][1] * natur_x[i][2])]
for i in range(len(natur_x))]

    natur_sq_x1 = [round(natur_x[i][0]**2, 3) for i in range(len(natur_x))]
    natur_sq_x2 = [round(natur_x[i][1]**2, 3) for i in range(len(natur_x))]
    natur_sq_x3 = [round(natur_x[i][2]**2, 3) for i in range(len(natur_x))]

    x1i = [norm_x[i][0] for i in range(n)]
    x2i = [norm_x[i][1] for i in range(n)]
    x3i = [norm_x[i][2] for i in range(n)]

    # b0, b1, b2, b3, b12, b13, b23, b123, b11, b22, b33
    list_for_b = [x0 + norm_x[i] + [norm_x12[i]] + [norm_x13[i]] + [norm_x23[i]]
+ [norm_x123[i]] + [norm_sq_x1[i], norm_sq_x2[i], norm_sq_x3[i]] for i in
range(n)]

    skm = linear_model.LinearRegression(fit_intercept=False)
    skm.fit(list_for_b, average_list_y)
    list_b = skm.coef_

    printed_matrixes()
    print("\nРівняння\n" + regression_str.format(*map(round_to_2, list_b)))
    cochrane_criteria(S_y)
    student_criteria(S_y, d)
    fisher_criteria(d)
#--------------------
m = 3
experiment(m)
```